1

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/039,789	01/02/2002	David K. Poulsen	INTL-0663-US (P12629)	9218

21906 7590 12/22/2006
TROP PRUNER & HU, PC
1616 S. VOSS ROAD, SUITE 750
HOUSTON, TX 77057-2631

EXAMINER	
YIGDALL, MICHAEL J	

ART UNIT	PAPER NUMBER
2192	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
2 MONTHS	12/22/2006	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

**BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

Application Number: 10/039,789
Filing Date: January 02, 2002
Appellant(s): POULSEN ET AL.

MAILED

DEC 22 2006

Technology Center 2100

E.E. Richards, II
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed on October 2, 2006 appealing from the Office action mailed on May 2, 2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct. No amendment after final has been filed.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

5,812,852	POULSEN et al.	9-1998
5,937,194	SUNDARESAN	8-1999
6,212,617	HARDWICK	4-2001

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

- Claims 1, 3, 6-8, 10-15, 18, 20, 21, 23 and 26 stand finally rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,812,852 to Poulsen et al. ("Poulsen") in view of U.S. Patent No. 5,937,194 to Sundaresan ("Sundaresan").

With respect to claim 1, Poulsen discloses a method comprising:

(a) receiving a first program unit in a parallel computing environment (see, for example, column 8, lines 29-30, which shows receiving a first parallel computer program unit).

Although Poulsen discloses that the first program unit includes parallel regions and global storage objects (see, for example, column 8, lines 29-30), Poulsen does not expressly disclose the limitation wherein the first program unit includes a reduction operation associated with a set of variables.

However, Sundaresan discloses a reduction operation associated with a set of values or variables, wherein the reduction operation performs an algebraic operation on the values or variables and is partitioned among a plurality of threads (see, for example, column 7, lines 13-16, and column 1, lines 59-63). The reduction operation in Sundaresan is encapsulated in reusable reduction objects so as to improve the expressibility and maintainability of parallel code (see, for example, column 5, lines 7-14, 21-23 and 30-33).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the system of Poulsen to include a reduction operation, such as with the reusable reduction objects taught by Sundaresan, for the purpose of improving the expressibility and maintainability of the parallel computer program. Furthermore, one of ordinary skill in the art would have been motivated to enhance the parallelism and performance of a parallel computer program (see, for example, Poulsen, column 1, lines 34-49) that includes such a reduction operation, given that the reduction operation is a candidate for this enhancement (see, for example, Sundaresan, column 10, line 60 to column 11, lines 4).

Therefore, Poulsen in view of Sundaresan discloses receiving a first program unit in a parallel computing environment, the first program unit including a reduction operation associated with a set of variables.

Poulsen in view of Sundaresan further discloses:

(b) translating the first program unit into a second program unit, the second program unit including a set of one or more instructions to partition the reduction operation between a plurality of threads including at least two threads and to reference a third program unit (see, for example, Poulsen, column 8, lines 32-35, which shows translating the program, and column 9, lines 2-12, which further shows translating parallel regions into library calls or a second program unit that references privatizable storage object declarations or a third program unit, and see, for example, Sundaresan, column 7, lines 13-16, which shows that the reduction operation is partitioned among a plurality of threads, as presented above); and

(c) translating the first program unit into the third program unit, the third program unit including a set of one or more instructions that encapsulate the reduction operation to perform an algebraic operation on the variables (see, for example, Poulsen, column 8, lines 32-35, which shows translating the program, and column 8, lines 59-61, which further shows translating global storage objects into privatizable storage object declarations or a third program unit that encapsulates the global storage objects, and see, for example, Sundaresan, column 7, lines 13-16, and column 1, lines 59-63, which shows that the reduction operation performs an algebraic operation on the values or variables, as presented above, and column 5, lines 7-14, which further shows that the reduction operation is encapsulated in a reduction object).

With respect to claim 3, Poulsen in view of Sundaresan further discloses reducing the set of variables logarithmically (see, for example, Sundaresan, column 7, lines 16-18, which shows that the reduction operation reduces the values or variables logarithmically).

With respect to claim 6, Poulsen in view Sundaresan further discloses associating the plurality of threads each with a unique portion of the set of variables (see, for example, Sundaresan, column 7, lines 20-21, which shows that the reduction operation associates individual values or variables to each of the threads).

With respect to claim 7, Poulsen in view of Sundaresan further discloses combining, in part, the variables associated with the plurality of threads in a pair-wise reduction operation (see, for example, Sundaresan, column 11, line 48 to column 12, line 7, which shows a sample reduction operation that combines the values or variables associated with the plurality of threads in a pair-wise reduction operation, wherein a given thread has a fan-in of two threads, which is to say a pair of threads).

With respect to claim 8, Poulsen discloses an apparatus comprising:

(a) a memory including a shared memory location (see, for example, column 8, lines 37-39, which shows a memory, and column 7, lines 7-10, which shows a global storage object in a shared memory location);

(b) a translation unit coupled with the memory (see, for example, column 8, lines 32-35, which shows a translation means).

Although Poulsen discloses a first parallel computer program unit (see, for example, column 8, lines 29-30) and further discloses translating the first program unit (see, for example, column 8, lines 32-35), Poulsen does not expressly disclose the limitation wherein the translation unit is to translate a first program unit including a reduction operation associated with a set of at least two variables into a second program unit, the second program unit to partition the reduction operation between a plurality of threads including at least two threads and to reference a third program unit, and wherein the translation unit is to also translate the first program unit into the third program unit, the third program unit to encapsulate the reduction operation to perform an algebraic operation on the variables.

However, Sundaresan discloses a reduction operation associated with a set of values or variables, wherein the reduction operation performs an algebraic operation on the values or variables and is partitioned among a plurality of threads (see, for example, column 7, lines 13-16, and column 1, lines 59-63). The reduction operation in Sundaresan is encapsulated in reusable reduction objects so as to improve the expressibility and maintainability of parallel code (see, for example, column 5, lines 7-14, 21-23 and 30-33).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the system of Poulsen to include a reduction operation, such as with the reusable reduction objects taught by Sundaresan, for the purpose of improving the expressibility and maintainability of the parallel computer program. Furthermore, one of ordinary skill in the art would have been motivated to enhance the parallelism and performance of a parallel computer program (see, for example, Poulsen, column 1, lines 34-49) that includes such a reduction operation, given that the reduction operation is a candidate for this enhancement (see, for example, Sundaresan, column 10, line 60 to column 11, lines 4).

Therefore, Poulsen in view of Sundaresan discloses a translation unit coupled with the memory, the translation unit to translate a first program unit including a reduction operation associated with a set of at least two variables into a second program unit, the second program unit to associate the reduction operation with one or more instructions operative to partition the reduction operation between a plurality of threads including at least two threads, the translation unit to also translate the first program unit into a third program unit, the third program unit to associate the reduction operation with a set of one or more instructions operative to perform an algebraic operation on the variables (see the rejection of claim 1 above).

Poulsen in view of Sundaresan further discloses:

(c) a compiler unit coupled with the translation unit and the memory, the compiler unit to compile the second program unit and the third program unit (see, for example, Poulsen, column

8, lines 42-45, which shows an executable program, which is to say a compiled program, and column 13, lines 11-13, which shows that the translation may be integrated with a compiler); and

(d) a linker unit coupled with the compiler unit and the memory, the linker unit to link the compiled second program unit and the compiled third program unit with a library (see, for example, Poulsen, column 8, lines 39-42, which shows a linker for linking the program with a library).

With respect to claim 10, Poulsen in view of Sundaresan further discloses the limitation wherein the variables in the set of variables are each uniquely associated with the plurality of threads and the library includes instructions to combine, in part, the variables associated with the plurality of threads (see, for example, Poulsen, column 10, lines 9-11 and 15-19, which shows that instructions in the library are called for each parallel region in the program, and Sundaresan, column 7, lines 20-21, which shows that the reduction operation associates individual values or variables to each of the threads).

With respect to claim 11, Poulsen in view of Sundaresan further discloses the limitation wherein the library includes instructions to combine, in part, the variables in a pair-wise reduction (see, for example, Sundaresan, column 11, line 48 to column 12, line 7, which shows a sample reduction operation that combines the values or variables associated with the plurality of threads in a pair-wise reduction operation, wherein a given thread has a fan-in of two threads, which is to say a pair of threads).

With respect to claim 12, Poulsen in view of Sundaresan further discloses a set of one or more processors to host the plurality of threads, the plurality of threads to execute instructions associated with the second program unit (see, for example, Poulsen, column 6, lines 46-50, which shows one or more processors for executing the plurality of threads).

With respect to claim 13, Poulsen in view of Sundaresan further discloses the limitation wherein the third program unit includes a callback routine and the callback routine is associated with instructions operative to perform the algebraic operation on at least two variables in the set of variables (see, for example, Poulsen, column 9, line 63 to column 10, line 9, which shows callback routines for the parallel regions in the program, and Sundaresan, column 7, lines 13-16, and column 1, lines 59-63, which shows that the reduction operation performs an algebraic operation on the values or variables).

With respect to claim 14, Poulsen in view of Sundaresan further discloses the apparatus of claim 13 wherein the library is operative to call the callback routine to perform, in part, a reduction on at least two variables in the set of variables (see, for example, Poulsen, column 10, lines 9-11 and 15-19, which shows that the routines in the library are called for each parallel region in the program, and Sundaresan, column 7, lines 13-16, which shows that the reduction operation performs a reduction on the values or variables).

With respect to claim 15, the limitations recited in the claim are analogous to the limitations recited in claim 1 (see the rejection of claim 1 above). Poulsen in view of Sundaresan further discloses a machine-readable medium that provides instructions, that when executed by a set of one or more processors, enable the set of processors to perform the recited method (see, for example, Poulsen, column 8, lines 37-39, and column 6, lines 46-50).

With respect to claim 18, see the rejection of claim 3 above.

With respect to claim 20, see the rejection of claim 7 above.

With respect to claim 21, Poulsen in view of Sundaresan further discloses performing a plurality of reduction operations in the third program unit (see, for example, Sundaresan, column 12, line 64 to column 13, line 2, which shows performing a plurality of reduction operations).

With respect to claim 23, Poulsen in view of Sundaresan further discloses a run-time library to implement the reduction operation (see, for example, Poulsen, column 9, line 63 to column 10, line 9, which shows a run-time library to implement the parallel operations).

With respect to claim 26, Poulsen in view of Sundaresan further discloses the limitation wherein the third program unit is to perform the algebraic operation using the library (see, for example, Poulsen, column 9, line 63 to column 10, line 9, which shows that the parallel operations are performed using the library).

- Claims 22, 24 and 25 stand finally rejected under 35 U.S.C. 103(a) as being unpatentable over Poulsen in view of Sundaresan, and further in view of U.S. Patent No. 6,212,617 to Hardwick ("Hardwick").

With respect to claim 22, Poulsen in view of Sundaresan does not expressly disclose performing a vector reduction operation in the third program unit via a N-dimension loop in the third program unit.

However, Hardwick discloses reduction operations that are applied to vectors formed from the basic data types (see, for example, column 6, lines 27-39). The vector reduction operations are performed via loops in such a manner as to ensure portability across different parallel architectures (see, for example, column 6, lines 40-49).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the system of Poulsen and Sundaresan to perform reduction operations on vectors via loops in the third program unit, such as taught by Hardwick. One of ordinary skill in the art would have been motivated to extend the type-specific reduction operations of Sundaresan (see, for example, column 12, lines 10-15) to further include vectors formed from the basic data types.

With respect to claim 24, although Sundaresan discloses performing a plurality of reduction operations (see, for example, column 12, line 64 to column 13, line 2), Poulsen in view of Sundaresan does not expressly disclose the limitation wherein the third program unit is to perform a plurality of vector operations.

However, Hardwick discloses reduction operations that are applied to vectors formed from the basic data types (see, for example, column 6, lines 27-39). The vector reduction operations are performed in such a manner as to ensure portability across different parallel architectures (see, for example, column 6, lines 40-49).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the system of Poulsen and Sundaresan to perform reduction operations on vectors in the third program unit, such as taught by Hardwick. One of ordinary skill in the art would have been motivated to extend the type-specific reduction operations of Sundaresan (see, for example, column 12, lines 10-15) to further include vectors formed from the basic data types.

With respect to claim 25, Poulsen in view of Sundaresan does not expressly disclose the limitation wherein the third program unit is to perform a vector reduction operation via a N-dimension loop.

However, Hardwick discloses reduction operations that are applied to vectors formed from the basic data types (see, for example, column 6, lines 27-39). The vector reduction operations are performed via loops in such a manner as to ensure portability across different parallel architectures (see, for example, column 6, lines 40-49).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the system of Poulsen and Sundaresan to perform reduction operations on vectors via loops in the third program unit, such as taught by Hardwick. One of ordinary skill in the art would have been motivated to extend the type-specific reduction operations of Sundaresan (see, for example, column 12, lines 10-15) to further include vectors formed from the basic data types.

(10) Response to Argument

At the outset, it is noted that the term "program unit" recited in the claims is understood to mean merely some unit or some part of a program, such as "a collection of statements in a programming language" (specification, page 7, lines 13-15), and not necessarily a complete and separate program *per se*.

Argument A (brief, pages 12-14)

Appellant contends that Poulsen does not teach or suggest translating a first program unit into two different, additional program units (brief, page 12). Appellant focuses on FIG. 1 of Poulsen, which illustrates a translation means 120 translating a parallel computer program 100 into a second (translated) parallel computer program 130.

First, however, it is noted that FIG. 1 of Poulsen is analogous to Appellant's FIG. 2, which illustrates a translation unit 28 translating a first code 202 into a second code 204. To the extent that Poulsen's FIG. 1 shows one translation rather than two translations, as Appellant contends, it is nonetheless equivalent to Appellant's FIG. 2.

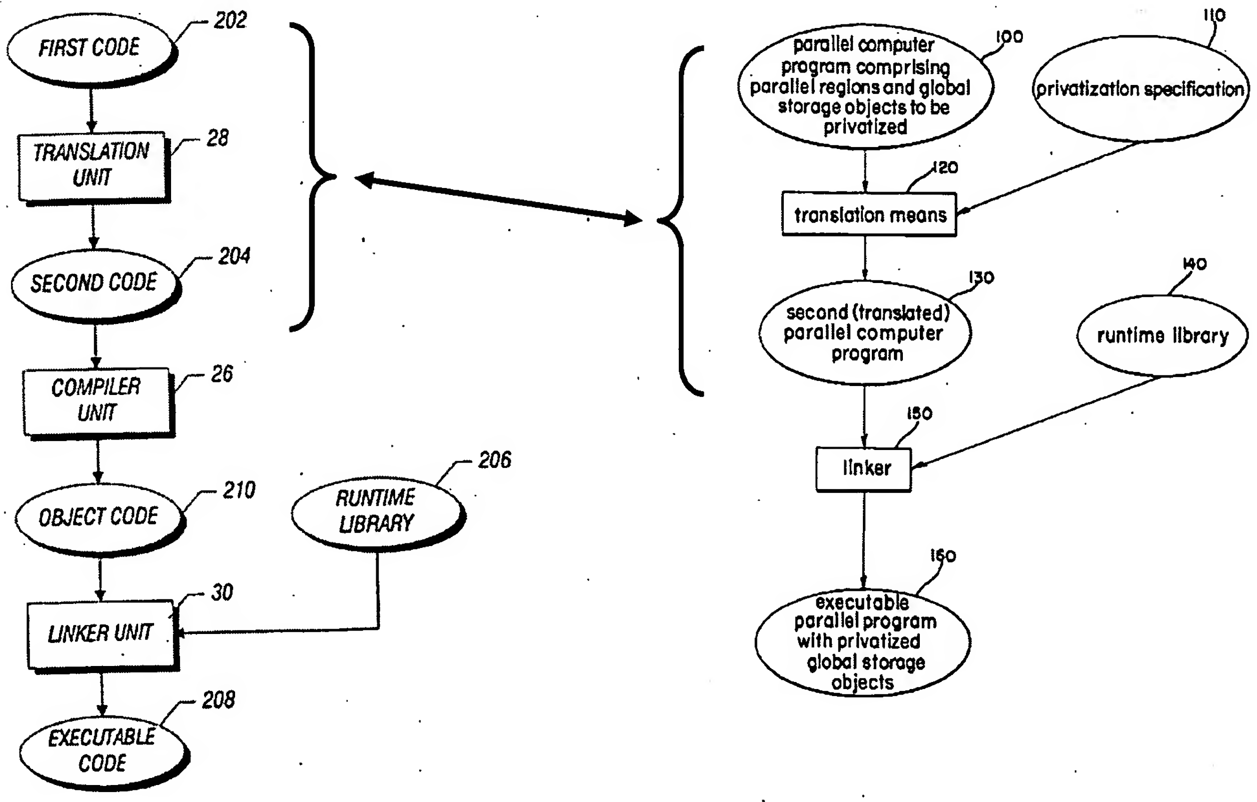
FIRST CODE
202
TRANSLATION UNIT
28
SECOND CODE
204
COMPILER UNIT
26
OBJECT CODE
210
RUNTIME LIBRARY
206
LINKER UNIT
30
EXECUTABLE CODE
208
parallel computer program comprising parallel regions and global storage objects to be privatized
100
privatization specification
110
translation means
120
second (translated) parallel computer program
130
runtime library
140
linker
150
executable parallel program with privatized global storage objects
160

FIG. 2 of Appellant

FIG. 1 of Poulsen

Second, it is noted that where the claim recites "translating the first program unit into a second program unit" and "translating the first program unit into a third program unit," the ordinal numbers recited in the claim are merely arbitrary labels. The examiner's interpretation is that the so-called second and third program units are included within Appellant's second code 204. For example, with reference to Appellant's FIG. 3, the translation unit 28 translates a program unit 301 into a call to a reduction routine 307 and a callback routine 305 (specification, page 7, lines 12-22). In FIG. 3, block 301 corresponds to the recited first program unit, block 307 corresponds to the recited second program unit, and block 305 corresponds to the recited third program unit (specification, page 8, lines 4-14). Appellant's argument seems to imply that a "program unit" is necessarily a complete and separate computer program, such that the translated program 130 of Poulsen could not include two different program units in the same manner as Appellant's second code 204. In this regard, Appellant's argument is contrary to the written description and drawings as originally filed, and improperly narrows the meaning of Appellant's own term "program unit" as noted above (specification, page 7, lines 13-15).

As noted above, a program unit is merely a collection of statements. Appellant's FIG. 4 illustrates examples of the recited second and third program units in blocks 403 and 405, respectively, and each program unit is a collection of statements (specification, page 8, line 27 to page 9, line 18). The second and third collections of statements result from the translation of first code 202 into second code 204 (specification, page 8, lines 15-19). Likewise, Poulsen teaches that two different collections of statements result from the translation of program 100 into translated program 130. Specifically, Poulsen teaches translating the program 100 into library calls (a second collection of statements) and privatizable storage object declarations (a third collection of statements) within the translated program 130 (column 8, lines 29-35 and 59-61, and column 9, lines 2-12). Here, in Poulsen, certainly program 100 is a first program unit, the library calls within translated program 130 are a second program unit, and the privatizable storage object declarations within translated program 130 are a third program unit. Thus, Poulsen teaches translating the first program unit into a second program unit and translating the first program unit into a third program unit, as recited in the claim.

The flow of operations within Appellant's translation unit 28 (FIG. 2) is illustrated in FIG. 3 (specification, page 7, lines 12-22). Likewise, in Poulsen, the flow of operations within translation means 120 (FIG. 1) is illustrated in FIG. 2 (column 8, lines 46-47). As illustrated below, the two figures show analogous operations. It is noted that in Appellant's FIG. 3, block 305 corresponds to the recited third program unit, and block 307 corresponds to the recited second program unit (specification, page 7, lines 12-22 and page 8, lines 4-14), as noted above. In Poulsen's FIG. 2, block 220 corresponds to the recited third program unit, and blocks 260 and 270 correspond to the recited second program unit. Again, the ordinal numbers recited in the claim are merely arbitrary labels. The labels do not reflect the sequence of operations illustrated in Appellant's FIG. 3, which instead shows the recited third program unit (block 305) ahead of the recited second program unit (block 307). Poulsen's FIG. 2 illustrates an equivalent sequence.

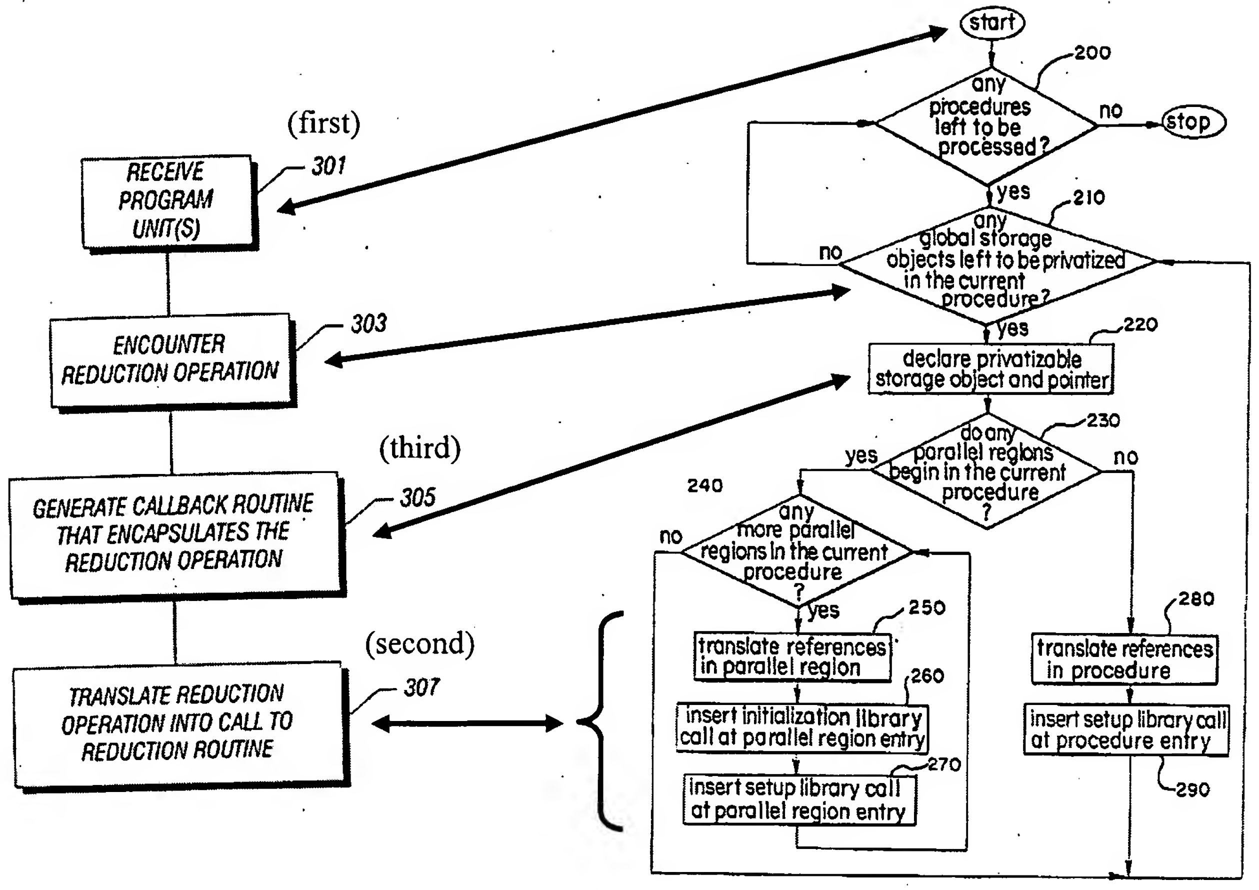
(first)
RECEIVE PROGRAM UNIT(S)
301
ENCOUNTER REDUCTION OPERATION
303
(third)
GENERATE CALLBACK ROUTINE THAT ENCAPSULATES THE REDUCTION OPERATION
305
(second)
TRANSLATE REDUCTION OPERATION INTO CALL TO REDUCTION ROUTINE
307
start
200
any procedures left to be processed?
no
stop
yes
210
any global storage objects left to be privatized in the current procedure?
no
yes
220
declare privatizable storage object and pointer
230
do any parallel regions begin in the current procedure?
yes
no
240
any more parallel regions in the current procedure?
no
yes
250
translate references in parallel region
260
insert initialization library call at parallel region entry
270
insert setup library call at parallel region entry
280
translate references in procedure
insert setup library call at procedure entry
290

FIG. 3 of Appellant

FIG. 2 of Poulsen

- block 305 – third program unit (specification, page 8, lines 4-9)
- block 307 – second program unit (specification, page 8, lines 10-14)

- block 220 – third program unit (column 8, lines 59-61)
- blocks 260, 270 – second program unit (column 9, lines 2-12)

Further analysis of Poulsen's teachings with respect to the claims is provided in response to Appellant's other arguments below.

Appellant contends that Poulsen does not teach or suggest that the library calls are instructions to partition a reduction operation between multiple threads, nor that the library calls reference a third program unit (brief, pages 12-13).

However, the rejection is based on a combination of Poulsen and Sundaresan. One cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

Poulsen discloses that the library calls (blocks 260, 270) are instructions to initialize parallel regions of the program (see, for example, column 9, lines 2-12). The parallel regions of the program comprise operations that are partitioned among multiple threads (see, for example, column 6, lines 46-54). Thus, the library calls are instructions to initialize parallel regions of the program and partition parallel operations among multiple threads. Furthermore, the library calls reference the privatizable storage object declarations (block 220). As noted above, the library calls correspond to the recited second program unit, and the privatizable storage object declarations correspond to the recited third program unit. Poulsen discloses that the library calls reference the privatizable storage object declarations, based on parameters input to the library calls, so as to allocate and initialize the privatizable storage objects (see, for example, column 9, lines 2-12, and column 9, line 63 to column 10, line 9). Thus, Poulsen teaches a second program unit (i.e., the library calls) that includes a set of one or more instructions to partition a parallel operation among a plurality of threads and to reference a third program unit (i.e., the privatizable storage object declarations).

Poulsen does not expressly disclose that the parallel operation is a reduction operation. Nonetheless, Sundaresan teaches a parallel operation partitioned among a plurality of threads that is, in fact, a reduction operation (see, for example, column 7, lines 13-18).

Appellant contends that the privatizable storage object declarations of Poulsen are not an encapsulation of a reduction operation, and that Poulsen does not teach or suggest performing an algebraic operation on a set of variables (brief, page 13).

However, as noted above, the rejection is based on a combination of Poulsen and Sundaresan. One cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

Poulsen discloses that the privatizable storage object declarations are instructions to encapsulate global storage objects (see, for example, column 8, lines 59-61). The global storage objects are accessed in the parallel regions of the program (see, for example, column 7, lines 7-10). As noted above, the parallel regions of the program comprise operations that are partitioned among multiple threads (see, for example, column 6, lines 46-54). Thus, Poulsen teaches a third program unit (i.e., the privatizable storage object declarations) that includes a set of one or more instructions to encapsulate a global storage object for performing a parallel operation on the global storage object.

Poulsen does not expressly disclose that the encapsulation is of a reduction operation for performing an algebraic operation on a set of variables. Nonetheless, Sundaresan teaches a reduction operation that performs an algebraic operation on a set of values (see, for example, column 1, lines 56-63). Sundaresan further discloses encapsulating the reduction operation in a reusable reduction object (see, for example, column 5, lines 7-14).

Appellant contends that there is no teaching or suggestion in the references to combine Poulsen and Sundaresan, and concludes that the combination is an improper hindsight-based reconstruction (brief, page 13). Appellant further speculates that the combination would change the principle of operation of Poulsen (brief, page 13).

However, the examiner submits that the record sets forth a *prima facie* case of obviousness. Sundaresan teaches encapsulating a reduction operation in a reusable reduction object so as to improve the expressibility and maintainability of parallel code (see, for example, column 5, lines 7-14, 21-23 and 30-33). Thus, it would have been obvious to a person having ordinary skill in the art to include such an encapsulated reduction operation in Poulsen, as Sundaresan suggests. Improving the expressibility and maintainability of a parallel computer program is an advantage of the reusable reduction object, and is therefore a reasonable teaching, suggestion or motivation to include the encapsulated reduction operation in Poulsen.

Moreover, the translation means 120 of Poulsen implements privatization to enhance the parallelism and performance of a parallel computer program (see, for example, column 1, lines 34-39). In Sundaresan, it is a parallel computer program that performs the reduction operation (see, for example, column 13, line 47 to column 14, line 3). Thus, it would have been obvious to a person having ordinary skill in the art to apply Poulsen's translation to a parallel computer program that includes a reduction operation. Enhancing the parallelism and performance of a parallel computer program is an advantage of Poulsen's translation, and is therefore a reasonable teaching, suggestion or motivation to apply the translation to Sundaresan's parallel computer program. Sundaresan even suggests that the reduction operation incorporates global storage objects suitable for privatization. Specifically, Sundaresan discloses that the reduction operation incorporates storage objects that are shared among a plurality of threads such that the storage objects are subject to data contention (see, for example, FIG. 3 and column 10, line 60 to column

11, line 4). Poulsen discloses that the privatization of such global storage objects would eliminate such memory access conflicts (see, for example, column 1, lines 34-49).

In response to Appellant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning (brief, page 13), it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from Appellant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971). Here, as noted above, both Poulsen and Sundaresan are concerned with parallel operations and encapsulation, and are therefore well suited for combination. The teachings, suggestions or motivations to combine the references are found within the references themselves.

Lastly, notwithstanding Appellant's statement that the combination would change the principle of operation of Poulsen (brief, page 13), there is no evidence to support Appellant's opinion that the combination would somehow change Poulsen's principle of operation.

<u>Argument B</u> (brief, page 14)

Appellant contends that neither Poulsen nor Sundaresan addresses using a run-time library to implement a reduction operation (brief, page 14).

However, the test for obviousness is not that the claimed invention must be expressly suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981).

Poulsen teaches using a run-time library to implement a parallel operation (see, for example, column 9, line 63 to column 10, line 9). Sundaresan teaches that the parallel operation is a reduction operation (see, for example, column 7, lines 13-18). Thus, the combined teachings of the references would have suggested to those of ordinary skill in the art using a run-time library to implement a reduction operation.

Argument C (brief, page 14)

Appellant contends that neither Poulsen nor Sundaresan addresses a third program unit to perform an algebraic operation using a library (brief, page 14).

Again, as noted above, the test for obviousness is not that the claimed invention must be expressly suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981).

Poulsen teaches performing a parallel operation using a library (see, for example, column 9, line 63 to column 10, line 9). Sundaresan teaches that the parallel operation is an algebraic operation (see, for example, column 1, lines 56-63). Thus, the combined teachings of the references would have suggested to those of ordinary skill in the art a third program unit to perform an algebraic operation using a library.

Argument D (brief, pages 14-15)

Appellant contends that Hardwick does not address a plurality of vector operations, and that none of the references teaches or suggests a third program unit to perform a plurality of vector operations (brief, page 14).

However, the multiple types of operations that Appellant notes in Hardwick are indeed vector operations. Sundaresan teaches performing a plurality of reduction operations (see, for example, column 12, line 64 to column 13, line 2), and Hardwick teaches reduction operations in the form of vector operations (see, for example, column 6, lines 27-39).

Appellant contends that there is less than no motivation for the combination of references with Hardwick (brief, page 15).

However, the vectors of Hardwick are formed from any basic data type or user-defined type (see, for example, column 6, lines 27-39), and Hardwick's vector operations are portable access different parallel architectures (see, for example, column 6, lines 40-49). Thus, it would have been obvious to a person having ordinary skill in the art to extend Sundaresan's type-specific reduction operations (see, for example, column 12, lines 10-15) to include vectors formed from any basic data type or user-defined type.

Furthermore, Hardwick discloses that the vector operations are included in parallel code that is converted and compiled for parallel execution in a manner comparable to Appellant's FIG. 2 and Poulsen's FIG. 1 (see, for example, column 4, lines 53-67). While Appellant argues that there is no teaching or suggestion to motivate one interested in performing program translations to implement the reduced inter-processor communications of Hardwick, reducing inter-processor communications, as Hardwick teaches in connection with the vector operations (see, for example, column 4, lines 35-52), is advantageous to any parallel architecture, such as the parallel architectures disclosed in Poulsen and Sundaresan.

Argument E (brief, page 15)

Appellant contends, as in Argument D above, that there is less than no motivation for the combination of references with Hardwick (brief, page 15).

However, as noted above, the vectors of Hardwick are formed from any basic data type or user-defined type (see, for example, column 6, lines 27-39), and Hardwick's vector operations are portable access different parallel architectures (see, for example, column 6, lines 40-49). Thus, it would have been obvious to a person having ordinary skill in the art to extend Sundaresan's type-specific reduction operations (see, for example, column 12, lines 10-15) to include vectors formed from any basic data type or user-defined type.

Furthermore, Hardwick discloses that the vector operations are included in parallel code that is converted and compiled for parallel execution in a manner comparable to Appellant's FIG. 2 and Poulsen's FIG. 1 (see, for example, column 4, lines 53-67). While Appellant argues that there is no teaching or suggestion to motivate one interested in performing program translations to implement the reduced inter-processor communications of Hardwick, reducing inter-processor communications, as Hardwick teaches in connection with the vector operations (see, for example, column 4, lines 35-52), is advantageous to any parallel architecture, such as the parallel architectures disclosed in Poulsen and Sundaresan.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Michael J. Yigdall
Examiner MY
Art Unit 2192

Conferees:

[illegible]
TUAN DAM
SUPERVISORY PATENT EXAMINER

Tuan Q. Dam, SPE 2192

Eddie C. Lee, TQAS 2100 [illegible]